ФПМИ БГУ

Вычислительные методы алгебры
Лаборатоная работа 4

Решение СЛАУ методом отражений

Подготовил:
Ткачук Павел
2 курс 1 группа

Преподаватель:
Будник Анатолий Михайлович

20 декабря 2016 г.

1 Постановка задачи

Система:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = b_n \end{cases} \tag{1}$$

Входные данные:

$$\left[\begin{array}{ccccc|c} 0.6444 & 0.0000 & -0.1683 & 0.1184 & 0.1973 & 1.2677 \\ -0.0395 & 0.4208 & 0.0000 & -0.0802 & 0.0263 & 1.6819 \\ 0.0132 & -0.1184 & 0.7627 & 0.0145 & 0.0460 & -2.3657 \\ 0.0395 & 0.0000 & -0.0960 & 0.7627 & 0.0000 & -6.5369 \\ 0.0263 & -0.0395 & 0.1907 & -0.0158 & 0.5523 & 2.8351 \end{array}\right]$$

Задача:

1. Методом отражений найти решение СЛАУ x
2. Найти вектор невязки $r = Ax - b$

2 Алгоритм

1. Полагаем $A^{(0)} = A$, $f^{(0)} = f$, $k = 1$
2. Умножаем систему слева на матрицу отражения V_k, получаем новую систему $A^{(k)}x = f^{(k)}$, где

   $$A^{(k)} = V_k A^{(k-1)},\ f^{(k)} = V_k f^{(k-1)}$$

   Матрицу V_k рассчитываем по формуле:

   $$\begin{aligned} &V_k = E - 2\omega^{(k)}(\omega^{(k)})^T, \text{ где} \\ &\omega^{(k)} = p^{(k)}(s^{(k)} - \alpha^{(k)}e^{(k)}), \\ &\alpha^{(k)} = \sqrt{(s^{(k)}, s^{(k)})}, \\ &p^{(k)} = \frac{1}{\sqrt{2(s^{(k)}, s^{(k)} - \alpha^{(k)}e^{(k)})}}, \\ &s_k = (0, \ldots, 0, a_{kk,k-1}, \ldots, a_{nk,k-1})^T \\ &e_k = (0, \ldots, 0, 1, 0 \ldots, 0)^T \text{(единица стоит на k-м месте).} \end{aligned}$$

3. Повторяем пункт 2 алгоритма, пока k не станет равным $n - 1$. В итоге приходим к системе

   $$A^{(n-1)}x = f^{(n-1)}$$

   в которой матрица $A^{(n-1)}$ являеться верхнетреугольной. Для нахождения неизвестных выполняем обратный ход, аналогичный обратному ходу метода Гаусса.

3 Результаты и вывод

3.1 Входные данные

0.6444 0.0000 -0.1683 0.1184 0.1973 1.2677
-0.0395 0.4208 0.0000 -0.0802 0.0263 1.6819
0.0132 -0.1184 0.7627 0.0145 0.0460 -2.3657
0.0395 0.0000 -0.0960 0.7627 0.0000 -6.5369
0.0263 -0.0395 0.1907 -0.0158 0.5523 2.8351

3.2 Выходные данные

```
Решаем матричное уравнение методом отражений
Основная матрица системы:
[[ 0.6444  0.     -0.1683  0.1184  0.1973]
 [-0.0395  0.4208  0.     -0.0802  0.0263]
 [ 0.0132 -0.1184  0.7627  0.0145  0.046 ]
 [ 0.0395  0.     -0.096   0.7627  0.    ]
 [ 0.0263 -0.0395  0.1907 -0.0158  0.5523]]
Свободные члены:  [ 1.2677  1.6819 -2.3657 -6.5369  2.8351]
Решение системы:  [ 0.99821505  1.99986528 -2.99975971 -9.00000843  6.00705353]
Невязка:  [  0.00000000e+00  -2.22044605e-16   0.00000000e+00   8.88178420e-16
   4.44089210e-16]
```

3.3 Вывод

Ответ с точностью до 5-ти знаков после запятой совпадает с ответом полученным методом Гаусса и методом квадратного корня.
Вектор невязки стал немного больше, в методах Гаусса и квадратного корня ни один элемент вектора невязки не превосходил $1e - 15$.

4 Листинг кода

```
import numpy as np
import numpy.linalg as linalg


# Метод отражений


def solve(A, f):
    size = len(f)
    E = np.eye(size)
    for i in range(size - 1):
        s = np.array([0 if j < i else A[j, i] for j in range(size)])
        e = E[:, i]
        a = np.sqrt(np.dot(s, s))
        k = 1/ np.sqrt(2 * np.dot(s, s - a * e))
        w = np.array(k * (s - a * e))
        V = np.array([[E[i, j] - 2 * w[i] * w[j] for j in range(size)] for i in range(size)])
        A = np.dot(V, A)
        f = np.dot(V, f)
    x = np.zeros(size)
    for i in reversed(range(size)):
        x[i] = (f[i] - sum(A[i, j] * x[j] for j in range(i + 1, size))) / A[i, i]
    return x


file = open("matrix", "r")  # Чтение файла
A, b = [], []
for line in file:
    A.append([float(el) for el in line.split()[:-1]])
    b.append(float(line.split().pop()))
A = np.array(A)
b = np.array(b)
print("Решаем матричное уравнение методом отражений")
print("Основная матрица системы:")
print(A)
print("Свободные члены: ", b)
ans = solve(A, b)
print("Решение системы: ", ans)
print("Невязка: ", (np.dot(A, ans) - b))
```